



ASHESI UNIVERSITY COLLEGE

DEVELOPING A WEB-BASED SYSTEM FOR INFERENTIAL DATA ANALYTICS

Applied Project

B.Sc. Computer Science

Abdul-Razak Adam

2018

ASHESI UNIVERSITY COLLEGE

Developing a Web-Based System for Inferential Data Analytics

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi
University College in partial fulfilment of the requirements for the award of
Bachelor of Science degree in Computer Science

Abdul-Razak Adam

April 2018

Declaration

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

.....

Candidate's Name:

.....

Date:

.....

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature:

.....

Supervisor's Name:

.....

Date:

.....

Acknowledgement

I would like to thank the Almighty Allah for seeing me through my four-year stay at Ashesi and for giving me the strength to complete this project. I would like to express my profound gratitude to everyone who supported me in my project especially my supervisors, Dr. Ayorkor Korsah, Dr. Hene Aku Kwapong and Mr. Arrojah Adade-Boafo for their feedback, critiques and time. My final thanks go to my family and friends for their motivation and positive energy which inspired me to work hard towards the completion of this project.

Abstract

The rapid generation of data by businesses has created a unique and exciting opportunity for management and employees to explore relationships in their data and make informed decisions based on these insights. However, limited sets of tools exist for businesses with no programming and data analysis skills for them to explore this opportunity. This paper proposes a new and improved web-based platform based on a desktop application, INFER, developed by Songhai Group. This application takes into consideration the features and limitations of software tools for data analytics that are currently available. The proposed solution is an end-to-end data analytics platform that enables users to build inferential models from observed data.

Table of Contents

Declaration	i
Acknowledgement	ii
Abstract	iii
Table of Contents	iv
List of Tables	vii
List of Figures.....	viii
Chapter 1: Introduction	1
1.1 Background	1
1.2 Problem	2
1.3 Objectives.....	2
1.4 Related Work.....	3
1.4.1 R	4
1.4.2 Orange	4
1.4.3 Weka.....	5
1.4.4 Datameer.....	5
1.5 Intended Solution.....	5
1.6 Benefits	6
Chapter 2: Requirement Specification	7
2.1 Project Scope.....	7
2.2 Project Perspective	7
2.3 System Functionality	8
2.4 Use-case	8
2.5 Use-case diagram.....	10
2.6 Operating Environment.....	10
2.7 Non-functional Requirements	11
2.7.1 Performance	11
2.7.2 Modularity	11
2.7.3 Scalability	11
2.7.4 Availability	11
2.7.5 Consistency	11
Chapter 3: Architecture	12

3.1 Architectural Considerations.....	12
3.2 Architectural Decisions.....	13
3.2.1 Client-Server Architecture.....	13
3.2.2 Model-View-Controller (MVC).....	13
3.3 System Context diagram.....	14
3.4 System Overview.....	15
3.4.1 Data Loading Module.....	16
3.4.2 Pre-processing Module.....	16
3.4.3 Modelling Module.....	16
3.4.4 Prediction Module.....	16
3.4.5 Results Module	16
3.5 System Architecture.....	17
Chapter 4: Implementation.....	18
4.1 Implementation Decisions.....	18
4.1.1 Choice of Programming Language	18
4.1.2 Decision on Framework	19
4.1.3 Choice of Framework.....	19
4.2 Project Setup	20
4.3 Technologies	20
4.3.1 Flask	20
4.3.2 SQLite.....	21
4.3.3 Docker	21
4.4 Environment.....	21
4.5 Running the Application	22
4.6 Flask Extensions	22
4.7 User Interface (UI) Dependencies	22
4.8 Implementation Structure.....	23
4.8.1 Initialization.....	23
4.8.2 Templates.....	24
4.8.3 Views Route functions	24
4.9 New INFER System	24
4.10 Implementation Summary.....	27
4.10.1 Loading Data Module.....	27
4.10.2 Pre-process Data Module.....	27
4.10.3 Build Model Module	27

4.10.4 Prediction Module	27
4.10.5 Result Module	28
Chapter 5: Testing and Results	29
5.1 Development Testing	31
5.1.1 Unit Testing	31
5.3 Component Testing	31
5.4 System Testing	32
Chapter 6: Conclusion	33
6.1 Recommendations and Future Work	33
6.2 Conclusion	34
References	35
Appendix A	37

List of Tables

TABLE 1.1: A SUMMARY OF RELATED TOOLS AND THEIR CHARACTERISTICS.	3
TABLE 3.1 : COMPARISON BETWEEN CLIENT-SERVER AND MODEL-VIEW-CONTROLLER ARCHITECTURE.....	14
TABLE 4.1 COMPARISON OF R, PYTHON AND JAVA.....	18
TABLE 4.2 COMPARISON BETWEEN DJANGO AND FLASK	20

List of Figures

FIGURE 2.1 USE CASE DIAGRAM OF THE NEW SYSTEM.....	10
FIGURE 3.1 SYSTEM CONTEXT DIAGRAM.....	15
FIGURE 3.2 SYSTEM OVERVIEW	15
FIGURE 3.3 HIGH-LEVEL SYSTEM ARCHITECTURE.....	17
FIGURE 4.1: HOMEPAGE OF THE NEW SYSTEM	25
FIGURE 4.2: STATISTICAL SUMMARY OF THE DATA.....	25
FIGURE 4.3: PRE-PROCESSING OPTIONS AVAILABLE IN THE NEW SYSTEM.....	26
FIGURE 4.4: LINEAR REGRESSION MODEL BUILD WITH THE NEW SYSTEM.....	26
FIGURE 5.1 PERFORMANCE OF THE NEW SYSTEM BY LIGHTHOUSE	30
FIGURE 5.2 PERFORMANCE DETAIL BY LIGHTHOUSE	30
FIGURE A.1: DOCKER CONFIGURATION FILE.....	37
FIGURE A.2: COMMAND TO BUILD DOCKER IMAGE.	37
FIGURE A.3: COMMAND TO RUN DOCKER CONTAINER.....	37
FIGURE A.4: RUNTIME DETAIL OF DOCKER CONTAINER.....	38
FIGURE A.5: LOGS OF DOCKER CONTAINER.	38
FIGURE A.6: INITIALIZATION FILE OF THE NEW SYSTEM.....	39
FIGURE A.7: BASE TEMPLATE EXTENDED BY ALL OTHER TEMPLATES.....	40
FIGURE A.8: ROUTE FUNCTION FOR UPLOADING DATA.	41
FIGURE A.9: SAMPLE UNIT TEST.	42
FIGURE A.10: HOMEPAGE OF OLD INFER DESKTOP APPLICATION.....	42
FIGURE A.11: SPECIFYING INPUT AND OUTPUT IN THE OLD INFER SYSTEM.....	42
FIGURE A.12: LEAST SQUARE REGRESSION RESULTS OF THE OLD INFER SYSTEM.	43

Chapter 1: Introduction

Since the early 2000s, the amount of data generated by businesses has risen sharply to several terabytes of data generated each day. Web applications, sensors, social media, devices and many other sources generate several terabytes of data. Analysing this data is crucial to management and employees in helping them make informed decisions, solve complex business problems and ultimately serve their customers better. This creates the need for tools to be developed to make it easy for non-experts in data science and programming to gain insights from their data. Songhai group¹ has developed INFER to meet this need. INFER is an offline PC-based program for building empirical models from a set of input variables and output variables. INFER was developed using C++ and Fortran. Project DAX seeks to migrate the INFER tool to a web-based platform and enhance its functionality. This current applied project is a subset of Project DAX that creates a proof-of-concept prototype of an online data analytics platform based on INFER.

1.1 Background

With incredible interest in solving critical and complex business problems with data, businesses all over the globe are using data analytics platforms to build and generate business models. These platforms also help businesses to explore relationships in their data and possibly make predictions from these models. An influencing factor has been the creation of new products and services, and new ways of doing business using data and machine learning, to learn trends, explore relationships on sales, cost and revenues. Because of these developments, businesses in Ghana are also beginning to take the idea of building out data analytics teams and platforms more seriously. This is aimed at managing and leveraging the large amount of data they create while doing business.

¹ <http://songhai.com/>

1.2 Problem

Though businesses in Ghana wish to leverage the potentials of such platforms to help improve their business models, it turns out that there are very limited set of tools for these businesses. The tools available are very complicated and require several hours of learning. For example, R is the most popular statistical package for analysing data; however, R requires users to know how to write code and have some data analysis skills. Most managers and employees however, do not have these skills. Therefore, there is the need for this project to explore the creation of an online data analytics platform for businesses and business professionals. This platform will help users explore relationships in their data, build empirical models from observed data and make predictions with the models.

1.3 Objectives

The goal of the overall project, Project DAX, is to migrate the functionality of the INFER to a web-based platform and enhance its functionality. However, the goal of the current applied project is to detail the functional requirements and architecture of the new system, and to implement a limited-functionality proof-of-concept prototype of the new system. A summary of objectives that Project DAX hopes to achieve are as follows:

- Build an online version of INFER that brings together a set of tools in a much more simplified form than it is usually found in tools like R, and with a business focus to support decision analytics for businesses.
- Enhance the functionality of the new systems to include common features found in current data analytics platforms.
- Based on the beta software from Songhai, Project DAX goal is to develop a full-fledged online application using latest web technologies and programming languages.

The current applied projects on the other hand hopes to achieve the following objectives:

- Develop a complete and modular system architecture. This includes: context diagram, system activities diagram and system architecture.
- Detail the functional and non-functional requirements of the new system.
- Implement a limited-functionality proof-of-concept prototype of the new system.

1.4 Related Work

In order to understand the current data analytics ecosystem, various tools for data exploration and analytics were studied. Table 1.1 presents an overview of tools explored. These tools were evaluated based on criteria such as platforms they run on, ease of usage, programming and data science skills requirements and other criteria.

Table 1.1: A summary of related tools and their characteristics.

	R	Orange	Weka	RapidMiner	KNIME	Datameer
Developer	R Core Team	University of Ljubljana	University of Waikato	RapidMiner	KNIME.com AG	Datameer
Licence	GNU GPL v2	GPL 3.0	GNU	AGPL	GNU	AGPL
Desktop / Web	Desktop	Desktop	Desktop	Desktop	Desktop	Desktop
Learning curve	Steep	Steep	Steep	Steep	Steep	Easy
Programming language(s)	R	Python	Java	Java	Java	Not specified
GUI / command line	GUI	Both	Both	Both	GUI	GUI
Main purpose	Scientific and Statistical computation	Data analysis	Data mining	Data analysis	Data analysis	Data analysis
Requires programming skills	Yes	No	No	No	No	No
Requires data science skills	Yes	Yes	Yes	Yes	Yes	Yes

1.4.1 R²

R is a programming language for statistical computation and analysis. It is the most popular statistical package among statisticians, students, researchers, data scientists and data miners (RDevelopment CORE TEAM, 2008). R unites all of the standard statistical tests, models, and analyses, as well as provides a full language for organizing and manipulating data (RDevelopment CORE TEAM, 2008). R comes with many packages for loading, cleaning, processing, analysing and visualizing data. R has packages that can be used to achieve all the functionality of this project, including loading data, cleaning, analysis and visualizing data. However, it has a very steep learning curve. As such, much of employees' time is focused on learning and getting immersed in the tool rather than spending time building insights to help create data proposals for their businesses. Another drawback for R is that it has many memory management issues (Ihaka & Gentleman, 1996). This issue is as a result of the core implementation of R that limits R to write all data on memory not disk (Venables, 2013). In other words, all data loaded in R is stored in the random-access memory of the computer therefore limiting the size of data that R can manipulate and analyse.

1.4.2 Orange³

Orange is an open source machine learning, data mining and analytics tool written mostly in Python and C++. Orange is developed for programmers, expert users, and students to explore and visualize the relationships in their data (Demšar et al, 2013). It was developed by the University of Ljubljana with the intention to be used for experimental data mining and data analytics. Orange makes analytics easier; however, like INFER, orange is a desktop application that currently runs on Windows, Mac OS and Linux.

² <https://www.r-project.org/>

³ <https://orange.biolab.si/>

Orange can be used for both simple and complex analytics like INFER. However, Orange has external functions that can be imported to extend its functionality.

1.4.3 Weka⁴

Weka is a Java application for data mining. It contains a collection of data mining and machine learning algorithms for general data mining tasks. Weka also provides a visualization toolkit to explore and analyse data (Rangra & Bansal, 2014).

1.4.4 Datameer

Datameer is an end-to-end data analytics platform for businesses. Datameer is built on top of Hadoop. Hadoop is an open source software utility by Apache Software Foundation that facilitates the use of multiple computers over a network. Hadoop is designed to solve problems involving massive amount of data.

1.5 Intended Solution

This project intends to research into latest web technologies including programming languages, web frameworks, databases, architectures and techniques to migrate the INFER tool to a web-based platform and enhance its functionality. The project is designed to understand the available technologies in data analytics for the web and generate a modular architecture for the new system. This proposed project combines the strengths of the available tools and libraries. It also provides a step-by-step approach to building inferential models. Moreover, it allows users to examine their data, screen the data for possible outliers or bad data points, edit the data, build a model, evaluate the model, and generate a predictive model for forecasts. The system will provide facilities to build both linear and non-linear models using techniques not limited to: simple linear

⁴ <https://www.cs.waikato.ac.nz/ml/weka/>

regression, stepwise regression, partial least squares regression and neural networks. The system will be modular on which other modules or components will be added.

1.6 Benefits

A prevalent problem with the current INFER system is that, it is a desktop application hence customers of Songhai have to go to the office of Songhai to use the application. This is inconvenient for both Songhai and its customers. For this reason, the INFER system is not available for use world-wide. This project seeks to make the functionality of the new INFER system available to all Songhai customers across the globe. This modular implementation of the new INFER will make it easy for more functionality to be added and maintained by other developers.

Chapter 2: Requirement Specification

The chapter provides a detailed description of the new INFER system. It covers both the functional and non-functional requirements of the system.

2.1 Project Scope

The new INFER system is a full-fledged and extensible web application currently consisting of five independent modules (data loading, pre-processing, modelling, prediction and results modules). The requirements of the new system are based on the requirements of the old system. The common features shared by the various tools presented in section 1.4 have been considered. Below is a list of general features that are common in most of the data analytics platforms studied in section 1.4:

- Most of these tools contain many machine learning algorithms for classification, regression, clustering and neural networks.
- All of the tools support data visualization. They support graphics such as scatter plots, histograms, box plots and 3D graphs.
- Each platform supports loading data from different sources including files (textual files and excel spreadsheets) and databases.
- The platforms support various pre-processing and transformation techniques such as normalization, discretization and principal component analysis.

2.2 Project Perspective

The new INFER system is a web-based product intended for use on web browsers of large screens such as laptops and tablets. It requires an internet connection to upload, process and analyse data. The system will be responsible for managing its database and

synchronizing its services. The system will also be responsible for handling client interactions, communicating data and displaying data requested by the client.

2.3 System Functionality

The new INFER provides a step-by-step approach to building inferential models. Inference modelling is the process of analysing and inferring descriptions and properties of data (Martin & Liu, 2013). The new system allows you to examine your data, screen the data for possible outliers or bad data points, edit the data, build and evaluate the model as well as make predictions based on the models generated. It provides an end-to-end data analysis experience to users. The functionality of the system as detailed earlier is influenced by the functionality of the old system and general features established among popular data analytics platforms. Figure A.10, A.11 and A.12 in Appendix A contain screenshots of the old INFER system.

2.4 Use-case

Use-case is a list of actions that defines the interaction of a user with the system. Below is a description of the users of the system and list of actions they can perform with the new system at each module.

Users of the system are called INFER clients. An INFER client will be responsible for carrying out all the functionality of the system. Below is a list of activities that a client can perform with the system.

- Upload data

Clients will be able to upload data in CSV or TRR format. TRR is the format for the input file for the previous INFER desktop application. The client will also be able to load data from databases and URLs.

- Pre-process / Clean data

Clients will be able to clean and pre-process the data they have uploaded to the system. Clients can perform the following cleaning methods:

- handle outliers,
- handle empty cells, and
- handle duplicates.

A client can also pre-process data to help generate more accurate models by:

- Scaling features to normalize and standardize the range of features of the data.

- Build models

For this version of the new system, clients will be able to build the following models:

- simple linear regression models,
- partial least square regression models,
- stepwise regression models, and
- neural net models.

Client will be able to regularize feature values to prevent models from overfitting. Overfitting arises when a model fits perfectly to the training dataset but performs poorly on the test set.

- Make Predictions

Clients will be able to make predictions based on any of the models they have built.

- Visualize and save the result of models

Clients will be able to visualize the results of models and the summaries of models' predictions on new sets of data. Clients will also be able to save these models for future use.

2.5 Use-case diagram

Use-case diagrams shows a graphical interaction of users of a system and the list of action users can perform on the system. Figure 2.1 is a use case diagram of the new system identifying how a client engages with the various components of the system.

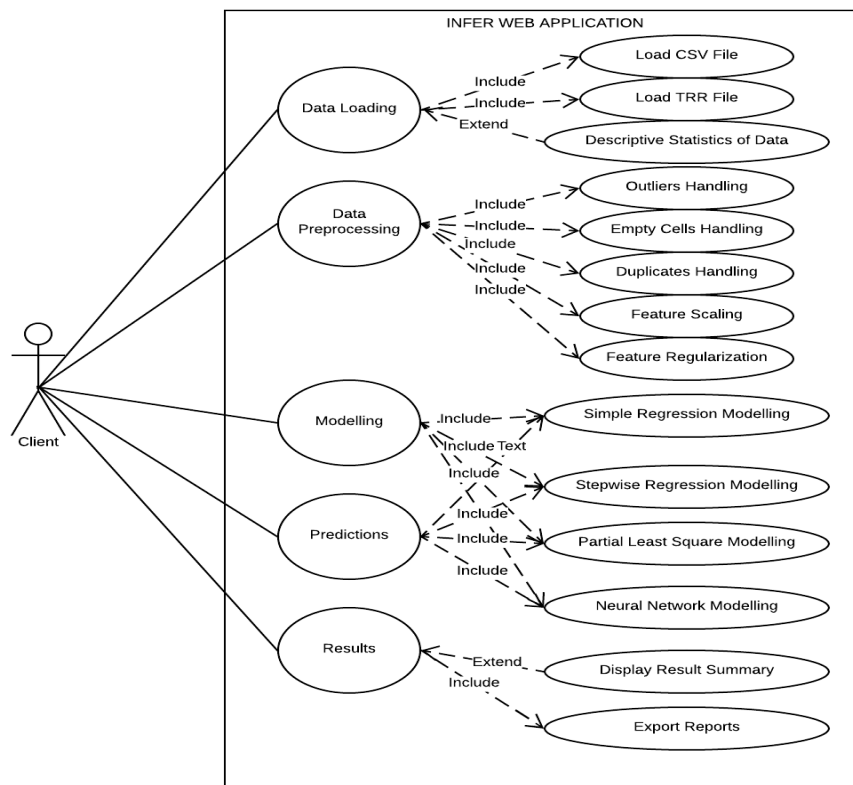


Figure 2.1 Use case diagram of the new system

2.6 Operating Environment

This section recommends the environment that supports the system. The system will be graphics and data-intensive; hence, it will require modern browsers such as Chrome build 56 or Firefox v56. Despite the fact that it will be responsive to various

screen sizes, larger screens are still recommended for clear visualizations and for full user experiences. The application will also rely on several open sources extensions.

2.7 Non-functional Requirements

These types of requirements are not directly concerned with the specified services that the system will provide to users (Sommerville, 2010). The following are the non-functional requirements of the system that were considered: performance, modularity, scalability, availability and consistency.

2.7.1 Performance: The new INFER system must be fast and must be able to produce the same or similar results as the old INFER system.

2.7.2 Modularity: The system should be subdivided into smaller and independent modules so that each module can be worked on, improved on and debugged separately. It should be modular so that new features can be added as modules and older modules can be updated and improved upon easily.

2.7.3 Scalability: The new system should be designed to scale easily. This means that the system should be able to accommodate an increasing number of users and requests without major impacts on its performance.

2.7.4 Availability: The system should be available to all customers of Songhai no matter where they are on the globe. They should be able to access the system once they have an internet connection.

2.7.5 Consistency: The new system should be consistent with the old system. The new system should have enhanced functionality, but it should provide consistent results and models as the old system.

Chapter 3: Architecture

This chapter discusses the high-level architecture of the new INFER system. It also discusses the various reasons for those decisions and the limitations of the chosen design paradigm.

3.1 Architectural Considerations

This section discusses the various considerations that influenced the design of the architecture of the new system. These considerations also influence the tools and technologies used.

- **Platform:** The new system will run on web browsers such as Google Chrome or Mozilla Firefox. This implies that the system will have to consider the capabilities and limitations of these browsers. With this, the project must be designed to cater for these constraints.
- **Scalability:** The system should be able to handle growing requests and clients without having very significant negative impact on its performance.
- **Performance at Scale:** Performance of the system must not reduce significantly as the number of workload and amount of data increases.
- **Security:** The new system should be secure, and user data should be protected at all cost. The system should employ the latest authentication and authorization systems.
- **Maintainability and Testability:** The code base should be very easy to understand and well-structured. Variables should be properly named to enhance debugging and enhancement.
- **Modularity:** The system should be divided into several subsystems such that each module is completely separable from other modules.

3.2 Architectural Decisions

In coming up with the architecture of the system, the various design considerations listed in section 2.7 were studied. These considerations influence the choices of software architecture considered. Software architecture shows a technical blueprint of how a software system is structured. Client-server and model-view-controller were considered because they are the recommended architectures for web applications (Sommerville, 2010). The following paragraphs discuss the two architectures.

3.2.1 Client-Server Architecture

In client-server architecture, functionality of the system is logically separated into services. Each service is delivered by an isolated server and clients make requests to these servers to use their services.

3.2.2 Model-View-Controller (MVC)

The MVC design pattern is a three-way factoring, in which the application is divided into three logical components; models, views, and controllers (Krasner & Pope, 1988). Models simulate the application domain, views display aspects of the model and controllers link user interactions from views with models. This design pattern increases modularity for simultaneous development and code reuse. MVC improves the scalability and maintainability of large applications by separating modules more intuitively (Krasner & Pope, 1988). Table 3.1 compares the two architectures.

Table 3.1 : Comparison between client-server and model-view-controller architecture.

	Client-Server Architecture	Model View Controller
Scalability	The services of the server and client can be easily enhanced since they are clearly separated.	Each module can be improved upon without affecting the other modules.
Maintainability	Maintenance is easy since services of the server and the clients are well separated.	Each module can easily be maintained because modules are independent of each other.
Modular	Client services are totally separated from server services.	Software is logically divided into three modules; models, views and controllers
Performance	Performance is unpredictable because it depends on the network connection.	MVC is generally fast since all interactions happen within a single server.
Security	The distribution of these services increases susceptibility of client-server to security threats.	All code is within one server making it easier to secure
Complexity	Client-server is logically easy to understand and requires no extra code for interactions between the server and the client.	MVC requires extra code for the interactions of the various models. This extra code could involve very complex relationships.

From the assessment in Table 3.1, both architectures meet the requirements of the system. However, MVC provides a better option in terms of security and performance. Therefore, the system architecture follows the MVC pattern.

3.3 System Context diagram

System context diagram defines the boundary of a system and its environment. It shows how entities in the environment interact with the system. Figure 3.1 shows how clients will interact with the new system.

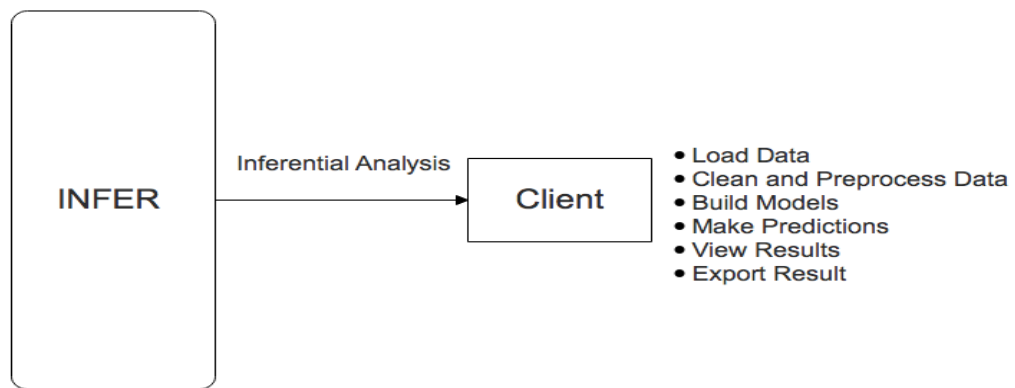


Figure 3.1 System context diagram

3.4 System Overview

The new system is divided into five modules. Each module interacts with the other modules. The diagram below shows the various modules that the INFER system is made of.

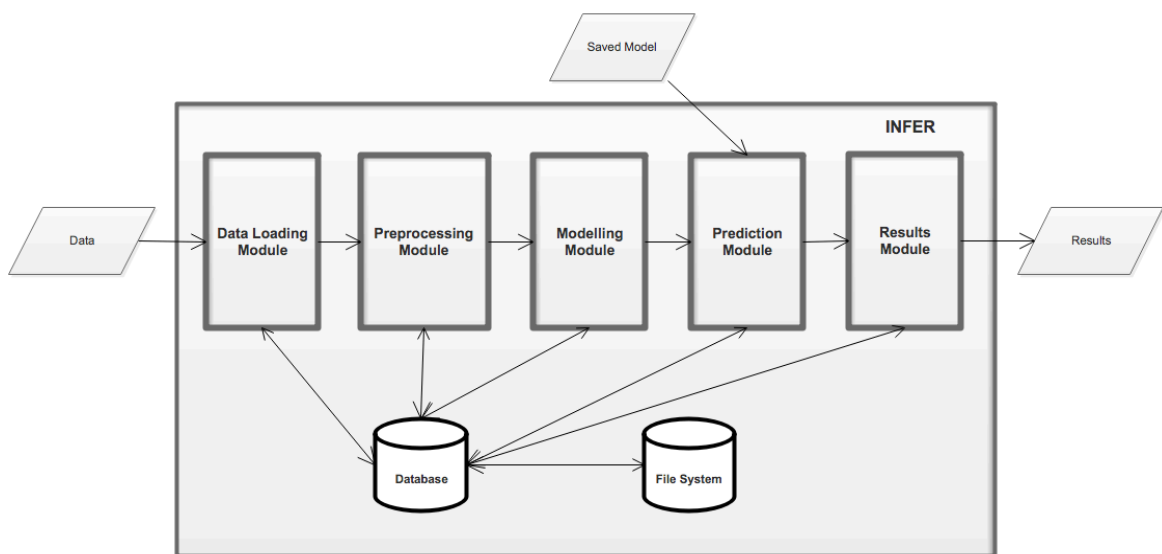


Figure 3.2 System overview

As shown in Figure 3.2, the new system is divided into five modules. Clients upload their data to the system through the data loading module, pre-process their data,

build models with this data and make predictions with models built. Client can then save their models using the result module. Each module is explained below.

3.4.1 Data Loading Module

The user loads data into the system through the data loading module. The data from this module serves as input to the pre-processing module.

3.4.2 Pre-processing Module

This module cleans the data uploaded by the user by handling duplicate rows, outliers, and empty cells. It also enables users to scale and regularize features for training to assist them build more accurate models. The cleaned data from this module serves as input to the modelling module.

3.4.3 Modelling Module

The user is given the option to build models including simple linear regression model, stepwise regression model, partial least squared regression or neural net models. This involves training the selected model and learning the right parameters to be used by the prediction module.

3.4.4 Prediction Module

This module allows a user to use parameters learned in the previous model to make projections on new data provided by the user.

3.4.5 Results Module

This module provides users options to save models generated in the form of an image, text file or excel spreadsheet. User can also extract pieces of code used in building these models.

3.5 System Architecture

Figure 3.3 shows the full architecture of the new INFER system. The architecture is divided into two sections. The external section and the internal section. The external section is made up of entities, systems, and technologies that are outside the INFER system such as users of the system, external service providers and web browsers.

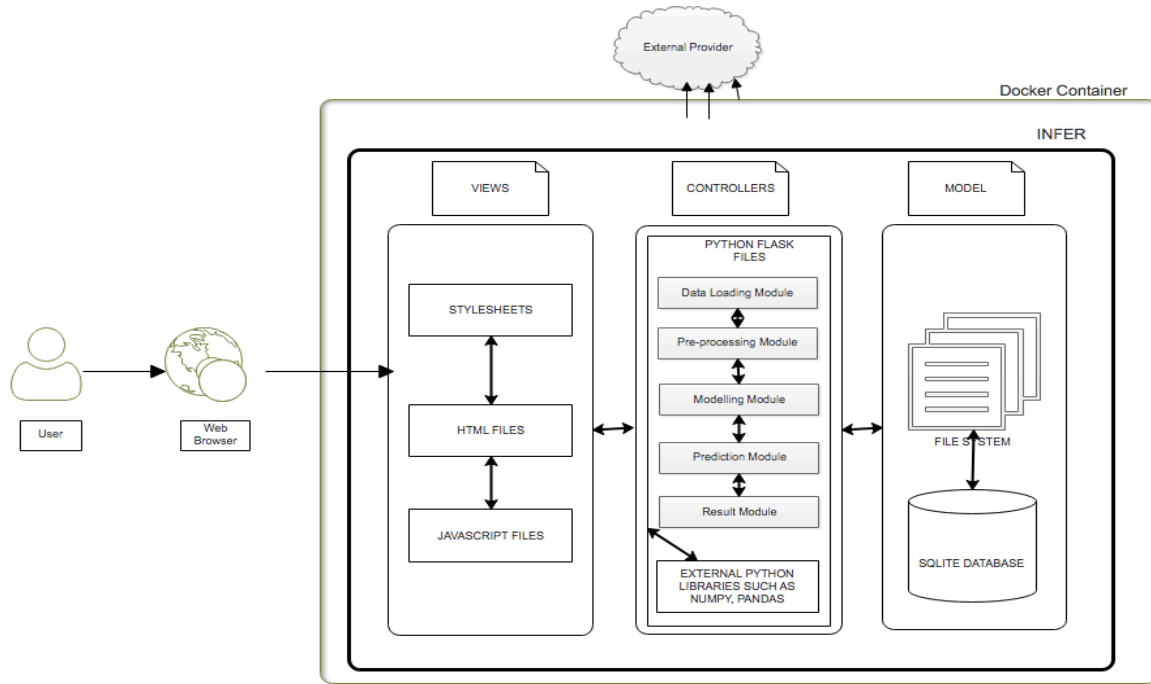


Figure 3.3 High-level system architecture

The internal section contains all the code, extensions, libraries, storage technologies that the system will use directly. The internal system is made up of models, views, and controllers. All modules of the system exist in the controller. The views of the system contain all user interfaces (UI) including HTML, CSS and all JavaScript files. The models contain the database objects of the system. The controllers initialize, set configurations for the application and defines routes. Controllers also render pages to users and map users' interactions with the models.

Chapter 4: Implementation

This chapter discusses the various implementation considerations that were taken into account and how they influenced the technologies used and the overall implementation of the system.

4.1 Implementation Decisions

In implementing the new system, several factors were taken into consideration such as the programming language and the framework within the programming language.

4.1.1 Choice of Programming Language

Three programming languages including R, Python and Java were considered because of their popularities in the field of data science and machine learning (Jovic, Brkic & Bogunovic, 2014). Table 4.1 shows the comparison of the three programming languages.

Table 4.1 Comparison of R, Python and Java

	Python	Java	R
Library for machine learning, data science and visualizations	Python has very matured libraries for machine learning, data analytics and visualization including numpy, pandas, scikit, SciPy, scikit-learn, TensorFlow and pyspark	Java has relatively smaller set of packages for data analytics and visualizations including Deeplearning4j, Weka and Java-ML.	R comes with very extensive packages for visualization, data analytics and machine learning including e1071, rpart and h2o.
Ease of usage	Python is very easy to learn and use.	Java requires more learning time and lots of code	R has a very steep learning curve
Syntax	Simple syntax	Complex and many syntaxes to get things done	Complex and compact syntax
Large Teams	Good for smaller teams and startups	Excellent for very large teams	Not good for large teams or projects since it lacks structure
Debugging	Easy to debug	Easy to debug	Easy to debug
Size of Codebase	Relatively smaller codebase	Relatively large code base due to the syntactic structure of Java	Relatively smaller codebase
Execution Time	Slower than Java	Faster execution of code	Slower than Java
Typing	Dynamic typing	Static typing	Dynamic typing

Verboseness	Simple and compact	Very verbose	Compact
Scalability	Scales easily	Scales easily	Very difficult to scale because it lacks structure.

From the comparison of the three programming languages, Python was selected because of its flexibility and its ability to scale easily. Python is also simple to learn, and it contains many packages for data science, machine learning and visualizations. However, Python is relatively slower than Java but many Python data analytics packages such as NumPy and TensorFlow are written in C and C++ which are equally as fast as Java.

4.1.2 Decision on Framework

The next decision after choosing the programming language, was to use a framework or to write vanilla code. A framework was considered because of the following benefits of web frameworks:

- Ease of development
- Increase speed of development
- Provide patterns for building scalable, maintainable and reliable applications.

4.1.3 Choice of Framework

The choice of framework was based on the following characteristics including flexibility, security, performance, extensions and scalability. Two of the most used python frameworks; Django and Flask were considered.

Table 4.2 Comparison between Django and Flask

	Django	Flask
Flexibility	Django is a large framework with many inbuilt extensions. This makes Django very rigid.	Flask is a microframework that gives users total control over the extensions they wish to include in their projects.
Security	It is reassuringly secure with several security measures in place such as SQL Injection and cross-site scripting.	Has a module to enable simple user authentication with similar security measures as found in Django
Performance	Very powerful	Very Powerful
Extensions	Similar extensions as Flask's	Similar extensions
Community	Large community support	Large community support
Ease of usage	Steep learning curve	Simple and easy to learn

Both frameworks are good candidates for implementing the system. However, Flask meets the needs of the system better because of its flexibility and simplicity. The rest of the chapter is influenced by the structure of Flask.

4.2 Project Setup

The new system is a Flask application that runs in a Docker container. The system makes use of various Flask extensions and UI dependencies.

4.3 Technologies

This section gives an overview of the various technologies used in developing the system. Each technology is briefly described and reasons for selecting these technologies are highlighted.

4.3.1 Flask ⁵

Flask is a python micro framework. It is a minimalistic framework for developing web applications. Rather than imposing development guidelines, Flask gives 100% flexibility to developers to add on any extension to their applications. This makes Flask applications scale easily and efficiently. Flask provides simplicity, flexibility, light-weighted and fine-grained control making maintenance, testing and scaling very easy

⁵ <http://flask.pocoo.org/>

(Grinberg, 2018). Flask also supports powerful and mature machine learning and data analysis libraries such as numpy, pandas, scipy, matplotlib, and scikit-learn.

4.3.2 SQLite⁶

The decision on the database to use was based on the needs of the platform. Speed and number of transactions were crucial features that were used to measure the various databases. Embedded databases are generally faster than any other types of databases since they do not make database transactions over a network. SQLite is currently the most mature and stable embedded database available (Owens & Allen, 2010).

SQLite is an embedded SQL database engine that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite requires no network or administration configuration (Owens & Allen, 2010). SQLite reduces overhead in terms of network calls and makes deployment of applications very easy.

4.3.3 Docker⁷

Docker is a container for bundling applications along with all of their dependencies. Docker enables the applications to run in different development, test and production environments without the need to set up a new environment (Merkel, 2014). Docker is used in this project to provide similar environments for development, testing, and production across teams.

4.4 Environment

A Docker container is set up to contain the environment of the project. For Docker to identify the application as a Docker application, the main directory of the application

⁶ <https://www.sqlite.org/index.html>

⁷ <https://www.docker.com/>

should contain a Docker file. The Docker file contains the configuration and command to run the application in the Docker container. Refer to Figure A.1 in Appendix A to view the Docker file of the project.

4.5 Running the Application

Appendix A contains information for setting up and running the new system in a Docker container. If set up successfully, the application can be accessed at the address <http://localhost:5000/>

4.6 Flask Extensions

Flask extensions are python modules that are added to a Flask application to extend the functionality of the application. Below is a list of flask extensions that are currently integrated in the system:

- Flask-SQLAlchemy⁸

This extension adds support of SQLAlchemy to the application. SQLAlchemy is an Object Relational Mapper (ORM) that provides an interface to interact with databases without the need to write SQL statements.

- Flask-Marshmallow⁹

Flask-Marshmallow is an object serialization/deserialization extension that protects the system from wrong and malicious inputs from users or attackers.

4.7 User Interface (UI) Dependencies

Various UI libraries have been integrated to enhance the usability and responsiveness of the system. The various UI dependences used in the system include:

1. Bootstrap¹⁰ version 4

⁸ <http://flask-sqlalchemy.pocoo.org/2.3/>

⁹ <https://flask-marshmallow.readthedocs.io/en/latest/>

Bootstrap is a modern web UI framework for developing responsive and mobile-first applications.

2. JQuery¹¹ version 3.2.1

JQuery is a JavaScript library that is required by Bootstrap. It helps with HTML elements manipulation.

3. Popper¹² version 1

It is a JavaScript tooltip required by Bootstrap for handling pop-ups.

4. DataTables¹³ version 1.10.16

DataTables is a JavaScript library that enables advanced interactive control on HTML table including searching, pagination, and design.

5. Shards¹⁴

It is a free modern UI toolkit built on top of Bootstrap version 4.

6. Awesome fonts¹⁵ version v5.0.8

It is a collection of beautiful icons for web and mobile applications.

4.8 Implementation Structure

This section gives an overview of the structure of the Flask project. It explains the various components of a Flask application.

4.8.1 Initialization

All Flask application must have an application instance. This application instance handles all requests between the client and the server using a Web Server Gateway

¹⁰ <https://getbootstrap.com>

¹¹ <https://jquery.com/>

¹² <https://popper.js.org/>

¹³ <https://datatables.net/>

¹⁴ <https://designrevision.com/downloads/shards/>

¹⁵ <https://fontawesome.com/>

Interface (WSGI) (Grinberg, 2018). WSGI is a python webserver for forwarding request to web applications. Figure A.6 in Appendix A contains the initialization file of the system.

4.8.2 Templates

Flask by default searches for HTML template in templates folder. Templates folder makes it easier for view functions to locate and render templates. Flask uses a Jinja2¹⁶ templating engine. A templating engine allows web developers to automate the generation of web pages. Jinja2 templates contain a combination of HTML code and Jinja2 code. Figure A.7 in Appendix A is the base template extended by other templates in the system.

4.8.3 Views Route functions

The application instance receives requests from the web browser hence it keeps URLs mapping functions called routes and handle each logic within that route function. The route view function showed in Figure A.8 of Appendix A handles the upload of data by a user to the system. If upload is successful, it redirects user to `screen_data.html` template. Otherwise, displays an error to the user.

4.9 New INFER System

This section describes the current state of implementation with supporting screenshots of the current system. Each screenshot is briefly described to aid understanding of the new system.

Figure 4.1 shows the first page that is displayed to the user. This page lists the data that the current user has worked with recently. The items in this list are ordered in terms of the last date accessed. In the screenshot shown, the user has worked with three data files and the last accessed file is the 'input_modified' file.

¹⁶ <http://jinja.pocoo.org/docs/2.10/>

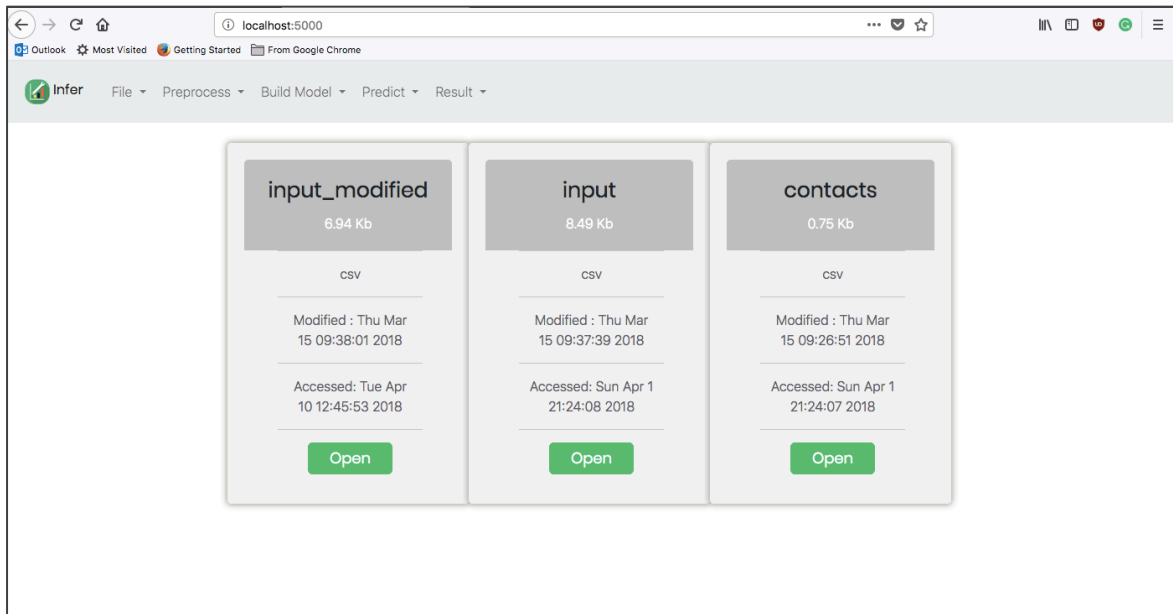


Figure 4.1: Homepage of the new system

One view in the system, shown in Figure 4.2, enables the user to view a statistical summary of the data. It shows the count, mean, standard deviation, minimum value, quartiles and maximum value of each column of the data.

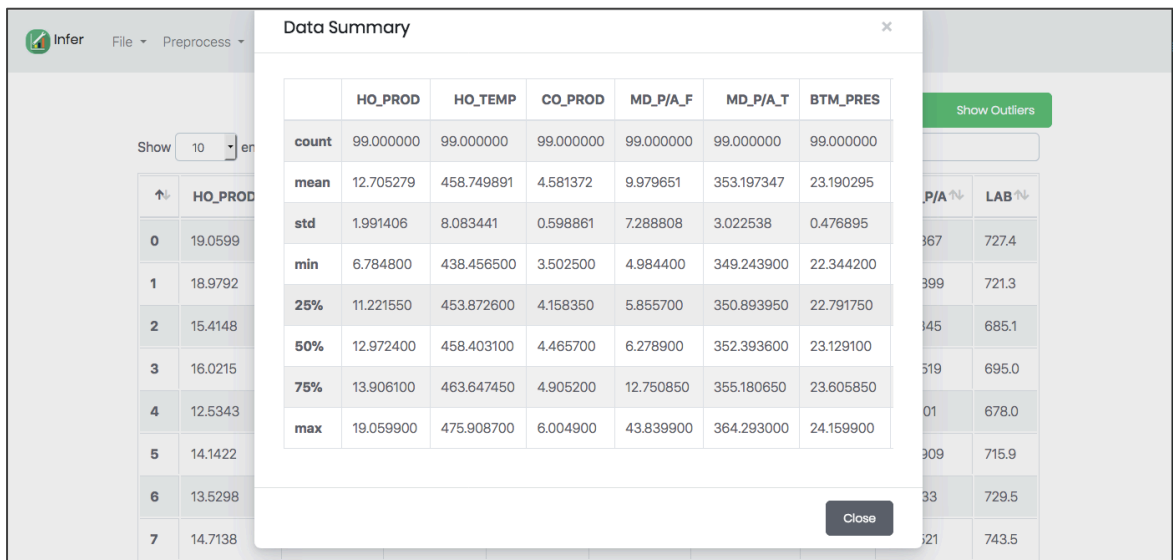


Figure 4.2: Statistical summary of the data.

Figure 4.3 illustrates how the user can select various options to enhance the accuracy of their models from Figure 4.3. Three options are currently available to users, but more options will be added in the next iteration of development.

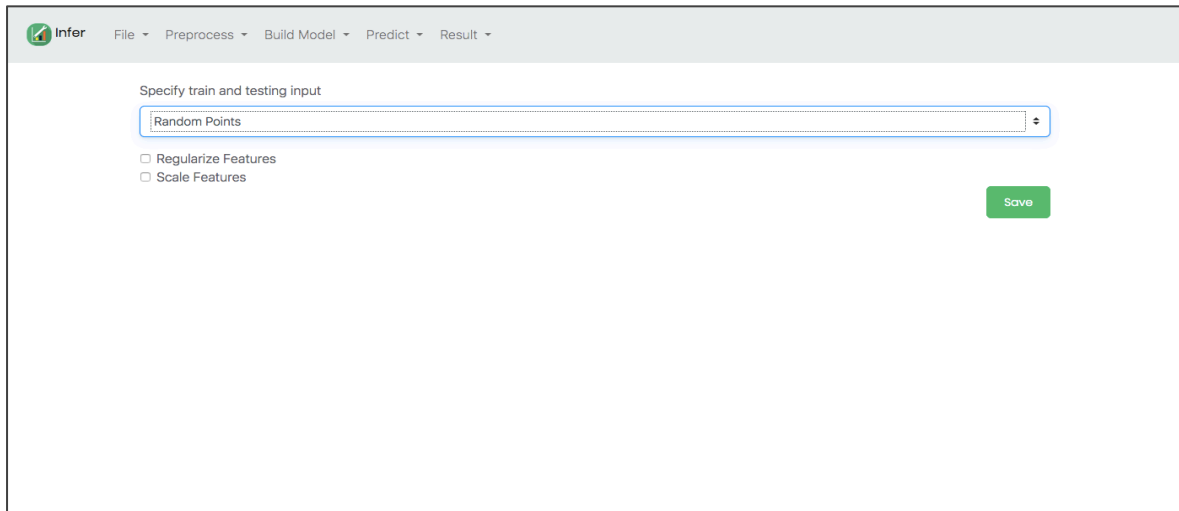


Figure 4.3: Pre-processing options available in the new system.

Figure 4.4 shows an example of a linear regression model that a user can build using the new system. Models built with the new system can be saved locally and loaded back into the system.

The screenshot shows the Infer software interface displaying the results of a linear regression model. The results are organized into several sections: a table of coefficients, a summary of model statistics, and diagnostic test results.

	coef	std err	t	P> t	[0.025 0.975]
const	3.8770	2.792	1.388	0.168	-1.666 9.420
HO_PROD	0.1835	0.025	7.283	0.000	0.134 0.234
HO_TEMP	-0.0035	0.006	-0.571	0.569	-0.016 0.009

Dep. Variable:	CO_PROD	R-squared:	0.361
Model:	OLS	Adj. R-squared:	0.348
Method:	Least Squares	F-statistic:	27.16
Date:	Wed, 18 Apr 2018	Prob (F-statistic):	4.49e-10
Time:	13:35:36	Log-Likelihood:	-67.017
No. Observations:	99	AIC:	140.0
Df Residuals:	96	BIC:	147.8
Df Model:	2		
Covariance Type:	nonrobust		

OLS Regression Results

Omnibus:	1.698	Durbin-Watson:	0.942
Prob(Omnibus):	0.428	Jarque-Bera (JB):	1.597
Skew:	0.306	Prob(JB):	0.450
Kurtosis:	2.891	Cond. No.	2.64e+04

Figure 4.4: Linear regression model build with the new system.

4.10 Implementation Summary

The system is implemented in five modules. Each module can scale horizontally and vertically. Vertical scaling means that more features can be added to modules and horizontal scaling means that previous features in each module can be updated without breaking the system. Each module contains its templates and view route functions.

4.10.1 Loading Data Module

This module currently supports uploading TRR and CSV files.

4.10.2 Pre-process Data Module

This module currently allows users to accomplish the following:

- Filling missing values by any of these three approaches
 - Filling missing values with zeros
 - Filling with the last value in that column or the next value in that column
 - Filling with the average of the column values
- Remove duplicates rows of the dataset
- Regularizing feature values to suppress less useful features and improve the accuracy of models.

4.10.3 Build Model Module

Training and testing of models are done in this module. The module currently supports building linear regression models.

4.10.4 Prediction Module

After parameters are learned, this module allows users to make predictions of their data with the models generated by the modelling module. Predictions for simple linear regression model is currently implemented.

4.10.5 Results Module

This module allows the user to save their models. Users can currently save their model as a file and read it in later.

Chapter 5: Testing and Results

Testing of the platform is intended to show that the system is doing what it is planned to do. Testing is also to ascertain defects and possibly correct these defects before the system is shared with customers of Songhai.

In order to calculate the performance, responsiveness, accessibility and best practices of the new system, Lighthouse¹⁷ by Google was used. Lighthouse audits the performance, accessibility, progressive web and many other pointers. Lighthouse can be accessed from the command line in chrome DevTools¹⁸. Chrome DevTools are developers' tools integrated in the chrome browser. DevTools can be accessed by right-clicking on any page in the chrome browser and by clicking on inspect.

The Lighthouse testing matrix include performance, accessibility, best practises and Search Engine Optimization (SEO). Performance is measured by how quickly contents of pages are visible and are usable to users. Accessibility measures how available the contents of the system are. Best practises measure how well the system follows guidelines of modern web development. SEO shows how well the system is optimized for search engines. Each matric is rated out of 100. Figure 5.1 shows the performance of the new system in Lighthouse.

¹⁷ <https://developers.google.com/web/tools/lighthouse/>

¹⁸ <https://developers.google.com/web/tools/chrome-devtools/>

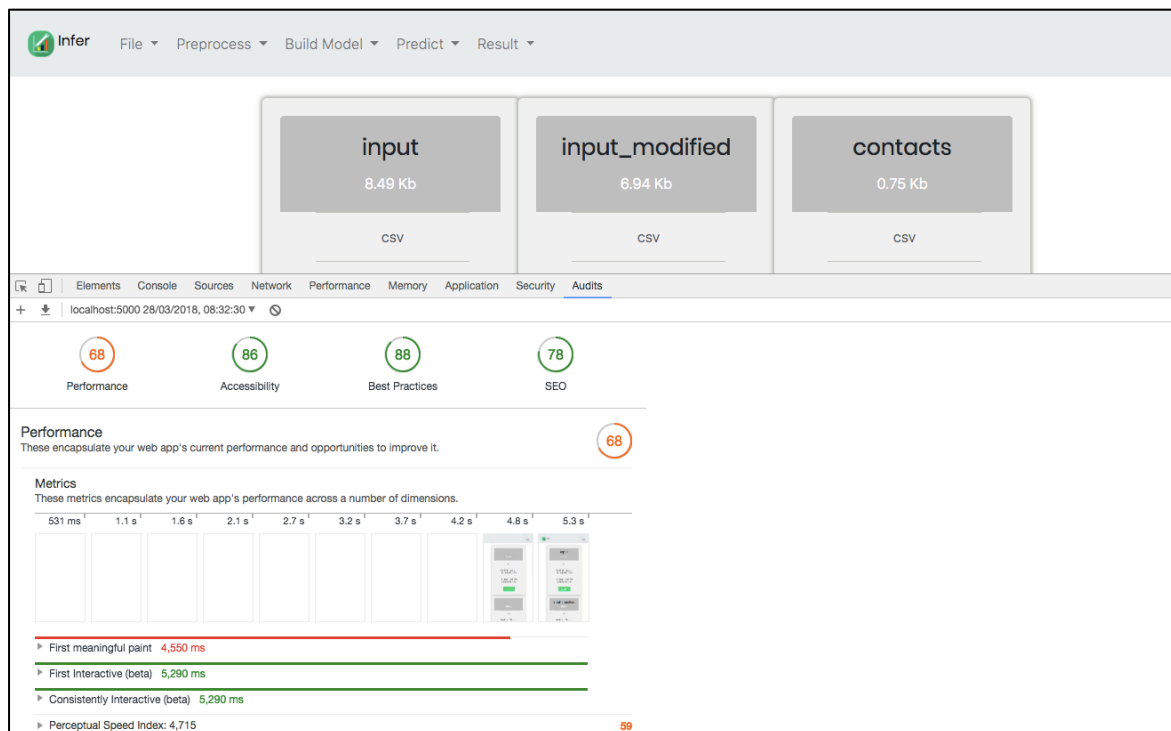


Figure 5.1 Performance of the new system by Lighthouse

Performance is currently 68% but there are lots of opportunities for performance to be increased. Figure 5.2 shows a list of these opportunities to improve the performance of the system.

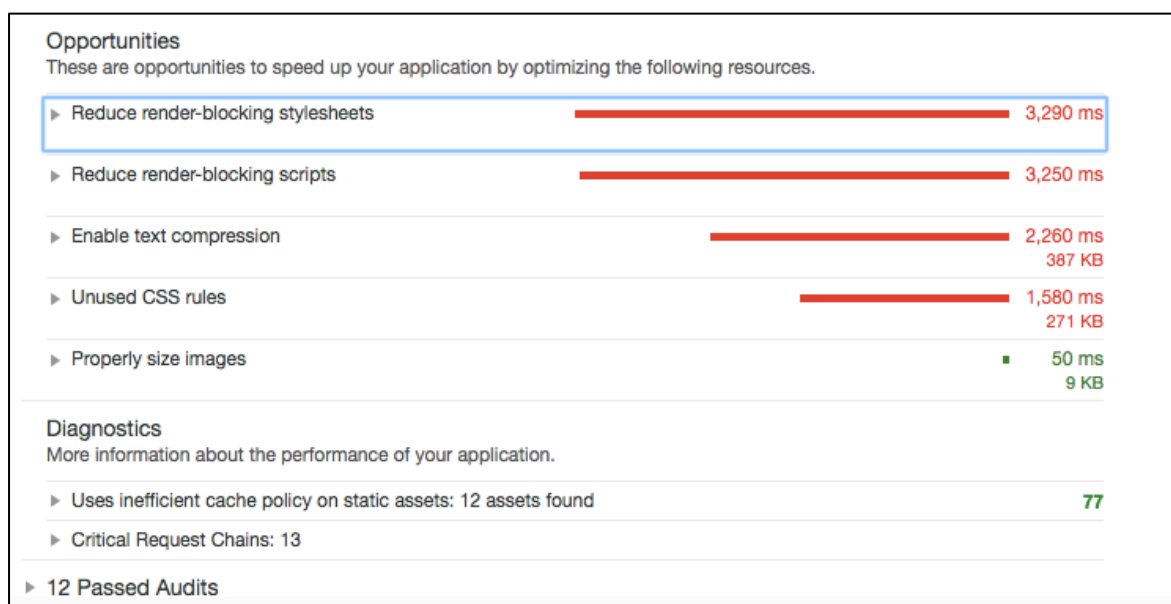


Figure 5.2 Performance detail by Lighthouse

Render blocking stylesheets and scripts are the highest performance issues that the system currently faces as suggested in Figure 5.2. This indicates that the browser is loading many CSS stylesheets and JavaScript scripts. A recommendation is to combine all stylesheets in a single file and all scripts into another file.

5.1 Development Testing

This type of testing is done by the developers of the system. Development testing consist of three levels including unit testing, component testing and system testing (Sommerville, 2010).

5.1.1 Unit Testing

Unit testing involves testing individual functions and classes. All functions within the system have been tested. The following functions were tested: index, screen_data, upload_file, edit_data, specify_input_output and linear_regression_model. Figure A.9 in Appendix A contains the unit test code for the upload_file function.

The unit test class in Flask sets up the application in testing mode. Individual unit test functions then assert contents or response of pages.

```
Ran 2 tests in 0.063s
OK
Process finished with exit code 0
```

Figure 5.2: Test result for sample unit test.

5.3 Component Testing

Component testing test involves testing each component or module of the system independently (Sommerville, 2010). Each of the five modules in the system including data loading, cleaning, building model and results modules have been tested individually and all work as expected.

5.4 System Testing

System testing involves testing the entire system by integrating the various components (Sommerville, 2010). All components of the system have been integrated successfully with each of the implemented components.

Chapter 6: Conclusion

This chapter gives an overview of the system in terms of implementation milestones and recommendations for further development.

6.1 Recommendations and Future Work

The new system can be made better and more usable by.

- Introducing a frontend framework or a library such as Angular ¹⁹ or React²⁰ as the system becomes more complex. This will make the code base of the system simpler to maintain and scale. It will also make the system more responsive.
- Adding support for loading in other files including Excel files, JSON and XML files. The system should also support loading data from external sources such as Microsoft Azure and Google clouds.
- Adding options to allow user to scale, regularize and normalize their data; this will help build more accurate models from user data.
- Handling duplicated row or columns and catering for missing values in data, these steps will ultimately help the system to build better and more accurate models from user data.
- Showing the distributions of each column of the data, and the scatter plot of all pair of columns in the data, this will help the user to better understand the distribution and relationships of their data, and hence help the user make better choices of models to build.
- Adding an option for user to specify the proportion of data to use as training and testing. This will give a better sense of control to users.

¹⁹ <https://angular.io/>

²⁰ <https://reactjs.org/>

- Making the system more user friendly; this initial stage of development did not concentrate much on the design and usability.
- Updating the menu titles to be more appropriate and more user-friendly titles.
- Implementing other linear and non-linear modelling methods such as partial least squares, step-wise regression and neural network modelling.
- Showing graphically the performance of models to both training and testing data. This will provide more insight to users and also make it easier for users to understand their models.

6.2 Conclusion

The new system provides an entire end-to-end data analytics work flow. It meets all the non-functional requirements and part of the functional requirements discussed in chapter 2. This project serves as a good start for other developers and designers to contribute and enhance the functionality of the system.

References

- Demšar, J., Curk, T., Erjavec, A., Gorup, Č., Hočevár, T., Milutinovič, M., & Štajdohar, M. (2013). Orange: data mining toolbox in Python. *The Journal of Machine Learning Research*, 14(1), 2349-2353.
- Grinberg, M. (2018). *Flask web development: developing web applications with python*. "O'Reilly Media, Inc."
- Ihaka, R., & Gentleman, R. (1996). R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3), 299-314.
- Jovic, A., Brkic, K., & Bogunovic, N. (2014). An overview of free software tools for general data mining. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on* (pp. 1112-1117). IEEE.
- Krasner, G. E., & Pope, S. T. (1988). A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object-oriented programming*, 1(3), 26-49.
- Martin, R., & Liu, C. (2013). Inferential models: A framework for prior-free posterior probabilistic inference. *Journal of the American Statistical Association*, 108(501), 301-313.
- Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239), 2.
- Owens, M., & Allen, G. (2010). *SQLite*. Apress LP.
- Rangra, K., & Bansal, K. L. (2014). Comparative study of data mining tools. *International journal of advanced research in computer science and software engineering*, 4(6).
- RDevelopment CORE TEAM, R. (2008). R: A language and environment for statistical computing.

Sommerville, I. (2010). *Software engineering*. New York: Addison-Wesley.

Venables, B. (2013). Stored Object Caches for R.

Appendix A

Docker configuration file

```
# this is an official Python runtime, used as the parent image
FROM python:3.6.4-slim

# set the working directory in the container to /app
WORKDIR /app

# add the current directory to the container as /app
ADD . /app

# pip command, pip install -r
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# unblock port 5000 for the Flask app to run on
EXPOSE 5000

# execute the Flask app
CMD ["python", "run.py"]
```

Figure A.1: Docker configuration file.

Figure A.1 instructs Docker to create a container with python version 3.6.4. Copy the application directory to WORKDIR used by Docker, then install the dependencies of the application and expose port 5000 for the application to run on. Finally, run the run.py file.

Building Docker image

```
$ docker build -t infer .      #infer is the name of the container
to build
```

Figure A.2: Command to build Docker image.

Running the container

```
$ docker run -d -p 5000:5000 infer
```

Figure A.3: Command to run Docker container.

This will start the application on port 5000

Runtime details of the container

```
$ docker ps
CONTAINER ID          IMAGE                  COMMAND               CREATED              STATUS              PORTS
e1db3bcb1513         infer                 "python run.py"      About
a minute ago         Up About a minute    0.0.0.0:5000->5000/tcp
heuristic_hopper
```

Figure A.4: Runtime detail of Docker container.

View logs of container

```
$ docker logs e1db3bcb1513
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 268-732-459
```

Figure A.5: Logs of Docker container.

Initialization file

This sets the initial configuration of the application.

```
'''
This file initializes the application and connects the app
instance to all views, models and controllers
'''
from flask_sqlalchemy import SQLAlchemy
from flask_marshmallow import Marshmallow
from flask import Flask

# file extensions to allow to be uploaded
#Add extension to allow in the list after csv
ALLOWED_EXTENSIONS = set(['csv', 'trr'])

# Initialize the app
app = Flask(__name__)

# load config from config.py
app.config.from_object('config')

# initialize db
db = SQLAlchemy(app)

# initialize serializer
ma = Marshmallow(app)

# load views
from app import controllers
```



```
# load_filters
from app import custom_filters
```

Figure A.6: Initialization file of the new system.

Base Template

This is a sample flask template that displays error messages to user.

```
<!doctype html>
<html lang="en">

<head>
    {% block head %}
        {#          title of the page specified by template
extending this template #}
        <title>{% block title %}{% endblock %}</title>
        <meta charset="utf-8">
        <base href="/">
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <link rel="icon" type="image/x-icon"
href="../../static/img/logo_edit.png">
        <link rel="stylesheet"
href="../../static/css/bootstrap.min.css">
        <link rel="stylesheet"
href="../../static/css/dataTables.bootstrap4.min.css">
        <link href="../../static/css/awesome-font.css"
rel="stylesheet">
        <link rel="stylesheet"
href="../../static/css/shards.min.css?version=2.0.1">
        <link rel="stylesheet" href="../../static/css/shards-
extras.min.css?version=2.0.1">

        <script src="../../static/js/jquery-
3.2.1.slim.min.js"></script>
        <script src="../../static/js/popper.min.js"></script>
        <script src="../../static/js/bootstrap.min.js"></script>
        {#          <script src="../../static/js/jquery-
1.12.4.js"></script>#}
        <script
src="../../static/js/jquery.dataTables.min.js"></script>
        <script
src="../../static/js/dataTables.bootstrap4.min.js"></script>

    {% endblock %}
</head>

<body>

{# message block for Flask to display global messages #}
{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
```

```

        {#          print all error message in the global
get_flashed_messages object #}
        {% for category, message in messages %}
            <div class="alert alert-{{ category }}">{{ message
}}</div>
        {% endfor %}
    {% endif %}
{% endwith %}

<div class="container">
    {# this is where content from any template extending this
class fits #}
    {% block content %}{% endblock %}
</div>
</body>

</html>

```

Figure A.7: Base template extended by all other templates.

Views Route functions

This route function allows user to upload data into the system.

```

@app.route('/open', methods=['GET', 'POST'])
def upload_file():
    '''
    this view basically handle the upload of data to the server.
    it protects from the upload of malicious files and save safe
    files to the upload folder defined in config.py
    :return: flask template of load_data.html
    '''
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash(u'No file part', alert_types['danger'])
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit a empty part without filename
        if file.filename == '':
            flash(u'No selected file', alert_types['danger'])
            # redirect to current url with an error message
            return redirect(request.url)
        if file and allowed_file(file.filename):

            # check if current file type is allowed
            filename = secure_filename(file.filename)

            # generate directory to store file
            directory = os.path.join(app.config['UPLOAD_FOLDER'],
filename)

```

```

        try:
            # check if file already in directory
            f = open(directory)
            f.close()
        except IOError:
            # if file not already in directory
            file.save(directory)

        # set global flask variable for working directory for
other views
        session['directory'] = directory

        flash(u'File uploaded successfully',
alert_types['success'])
        return redirect(url_for('screen_data',
directory=directory))
    else:
        error = 'File type not supported yet'
        flash(error, alert_types['danger'])
        return redirect(request.url)
    return render_template('load_data.html')

```

Figure A.8: Route function for uploading data.

Sample Unit test

Below is a sample unit test that was performed on the new system.

```

import unittest
from app_folder import app

class BasicTest(unittest.TestCase):

    def setUp(self):
        app.config['TESTING'] = True
        app.config['DEBUG'] = False
        self.app = app.test_client()
        # Disable sending emails during unit testing
        self.assertEqual(app.debug, False)

    def test_main_page(self):
        response = self.app.get('/', follow_redirects=True)
        self.assertEqual(response.status_code, 200)

    def test_home_data(self):
        # sends HTTP GET request to the application
        # on the specified path
        result = self.app.get('/open')
        # assert the response data
        self.assertIn(b'Select you data file', result.data)
        #self.assertEqual(result.data, "Select you data file")

if __name__ == "__main__":

```

```
unittest.main()
```

Figure A.9: Sample unit test.

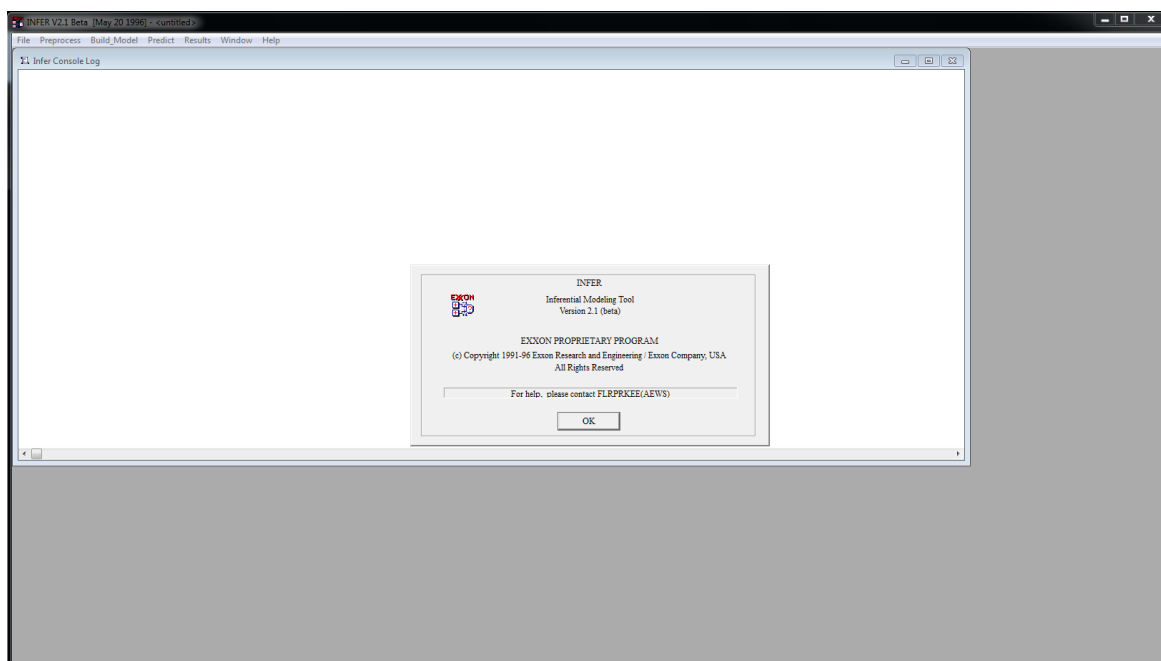


Figure A.10: Homepage of old INFER desktop application

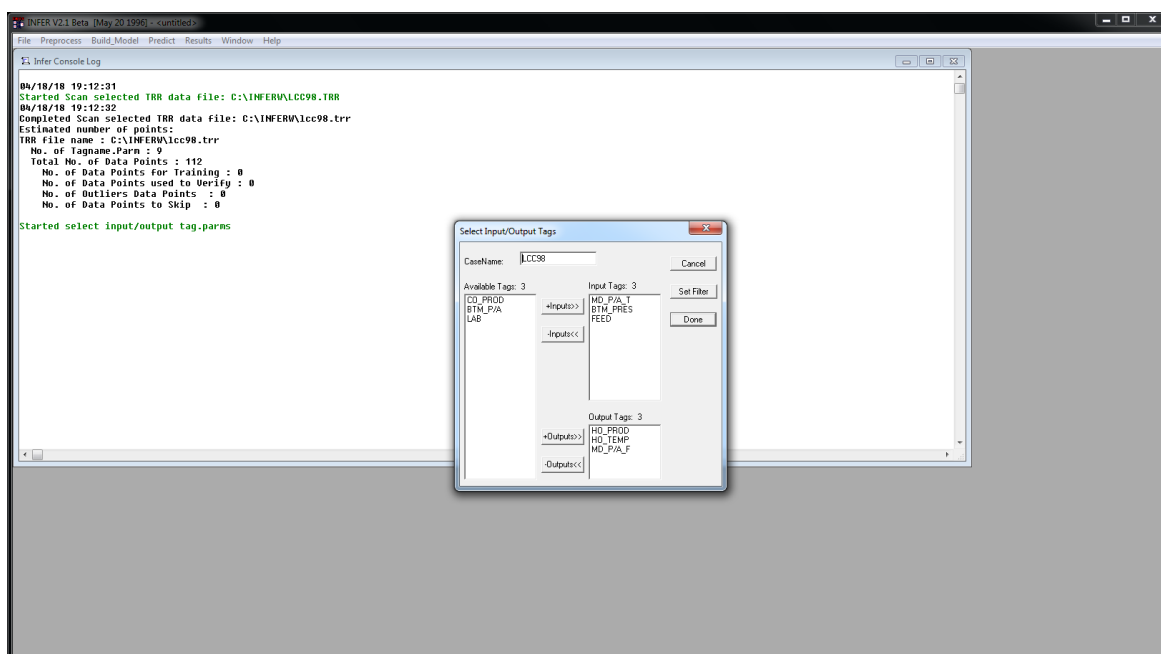


Figure A.11: Specifying input and output in the old INFER system

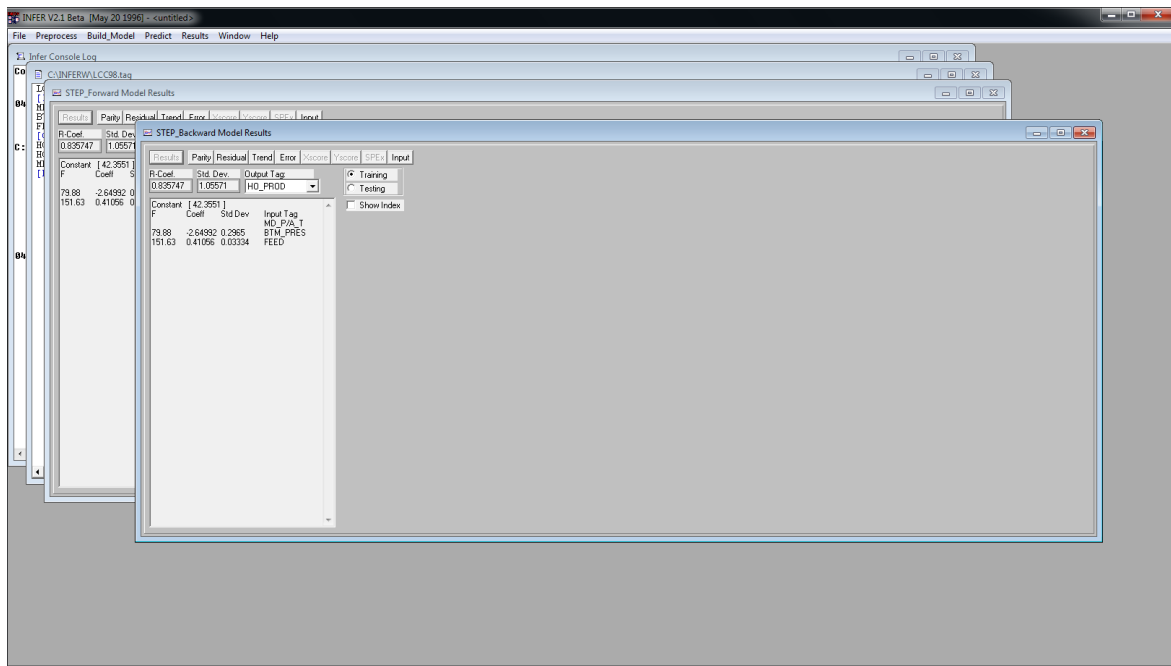


Figure A.12: Least square regression results of the old INFER system.